CLAIMS

1. A processor for executing a plurality of tasks each of which executes one or more computer instructions, wherein in normal operation the processor execution unit is to start execution of at most N instructions from any given task without starting execution of any intervening instruction from any other task, and after the N instructions the execution unit is to start execution of an instruction of another task if another task is available for execution.

2. The processor of Claim 1 wherein N = 1.

3. The processor of Claim 2 wherein the instruction execution is pipelined.

4. The processor of Claim 2 wherein each task performs processing on a data flow between networks, and after starting execution of any given instruction of any task that performs processing on any given data flow, the execution unit is to start executing an instruction of another task that performs processing on a different data flow if the other task is available for execution.

5. A method for executing a plurality of tasks, the method comprising:

an execution unit starting execution of at most N instructions from any given task without starting execution of any instruction from any

other task, wherein N is a predetermined number; and

after starting execution of at most N instructions from any given task, starting execution of an instruction of another task if another task is available for execution.

6. The method of Claim 5 wherein N = 1.

7. A multi-tasking computer processor which includes, for each task, one or more registers storing task-specific values, such that no one of the one or more registers has to be saved or restored when a task is scheduled for execution.

8. The processor of Claim 7 wherein the one or more registers include a program counter register for each task.

9. The processor of Claim 7 wherein the tasks are subdivided into sets of one or more tasks each, and for each set the processor includes one or more registers for storing task-specific values of the tasks of the set.

10. A method for executing a plurality of tasks by a computer processor, the method comprising executing tasks, wherein tasks use one or more registers storing task-specific values but different tasks use different ones of the one or more registers,

so that interrupting execution of one task and starting execution of another task does not involve saving values of the one or more registers or restoring values of the one or more registers.

11. In a multi-tasking computer system comprising a plurality of resources to be shared by a plurality of tasks, a circuit for allocating each resource to the tasks so that after any one of the tasks has finished accessing any one of the resources in processing a data unit, the task does not get access to the same resource until after every other one of the tasks has finished accessing the resource.

12. The circuit of Claim 11 wherein for at least one resource, each task starts accessing the resource by locking the resource to make it unavailable to any other task, and the task finishes accessing the resource by unlocking the resource.

13. A method for sharing a plurality of resources by a plurality of computer tasks, the method comprising:

allowing a task T1, which is one of the tasks, to access all of the resources, and disallowing any other task from accessing any one of the resources; and

for each resource, after the task T1 has finished accessing the resource, allowing another task to access the resource, and disallowing the task T1 from

accessing the resource until every other task sharing the resource has finished accessing the resource.

14. A processor for executing instructions such that when the processor executes a first instruction accessing an unavailable resource, the processor suspends the first instruction and the processor circuitry which was to execute the first instruction becomes operable to execute one or more other instructions.

15. The processor Claim 14 wherein the processor executes the first instruction to completion when the resource becomes available.

16. The processor of Claim 14 wherein when the first instruction becomes suspended, the first instruction is canceled, and the first instruction is re-executed when the resource becomes available.

17. The processor of Claim 14 wherein the processor performs multi-tasking, and a task executing the first instruction becomes suspended when the first instruction is suspended, and while the task is suspended the processor circuitry that was to execute the first instruction is operable to execute one or more other tasks.

18. A multi-tasking processor comprising task scheduling circuitry, such that when a task TA1

executed by the processor attempts to access an unavailable resource, the task scheduling circuitry suspends the task TA1 at least until the resource becomes available, and if another task TA2 is ready for execution in place of the task TA1 when the resource is unavailable to the task TA1, the task scheduling circuitry schedules the task TA1,

wherein the task scheduling circuitry operation does not involve instruction execution by the processor.

19. A multi-tasking processor comprising:

first circuitry for generating a first signal indicating whether a task suspend condition is true; and

second circuitry for scheduling a task or tasks for execution in response to the first signal.

20. The processor of Claim 19 further comprising third circuitry for generating a release signal indicating whether a release condition is true for releasing a task from the suspend condition,

wherein the second circuitry is responsive to the release signal when the second circuitry schedules a task or tasks for execution.

21. The processor of Claim 19 further comprising, for each task, a separate circuit for generating a signal SIG1 indicating whether the task is ready for

execution, wherein the second circuitry is responsive to one or more signals SIG1 in scheduling a task or tasks for execution.

22. The processor of Claim 19 wherein the second circuitry is to schedule a task or tasks for execution on each instruction executed by the processor such that whenever the processor is to execute any instruction, the second circuitry is to perform the task scheduling to schedule a task that will execute the instruction.

23. A method for executing computer instructions, the method comprising:

executing a first instruction accessing a computer resource;

if the resource is unavailable, then suspending the first instruction and executing one or more other instructions by circuitry which was to execute the first instruction.

24. The method of Claim 23 further comprising executing the first instruction to completion when the resource becomes available.

25. The method of Claim 23 wherein executing the first instruction comprises executing the first instruction by a first task, and

when the first instruction is suspended, the first task is suspended and execution of the one

or more other instructions comprises execution of one or more other tasks.

26. A multi-tasking method comprising:

generating a first signal indicating whether a task suspend condition is true;

scheduling a task or tasks for execution in response to the first signal.

27. The method of Claim 26 further comprising generating a release signal indicating whether a release condition is true for releasing a task from the suspend condition,

wherein scheduling a task or tasks for execution is responsive to the release signal.

28. The method of Claim 26 further comprising generating, for each task, a separate signal indicating whether the task is ready for execution.

29. The method of Claim 26 wherein scheduling a task or tasks for execution is performed on each instruction executed by any one of the tasks such that whenever an instruction is to be executed, the task scheduling is performed to schedule a task that will execute the instruction.